Reading Web Pages

The main function of a browser is to present the web resource you choose, by requesting it from the server and displaying it in the browser window. The resource is usually an HTML document, but may also be a PDF, image, or some other type of content. The location of the resource is specified by the user using a URI (Uniform Resource Identifier).

The rendering engine will start getting the contents of the requested document from the networking layer. This will usually be done in 8kB chunks.
After that, this is the basic flow of the rendering engine:

Figure : Rendering engine basic flow

The rendering engine will start parsing the HTML document and convert elements to DOM nodes in a tree called the "content tree". The engine will parse the style data, both in external CSS files and in style elements. Styling information together with visual instructions in the HTML will be used to create another tree: the render tree.
The render tree contains rectangles with visual attributes like color and dimensions. The rectangles are in the right order to be displayed on the screen.
After the construction of the render tree it goes through a "layout" process. This means giving each node the exact coordinates where it should appear on the screen. The next stage is painting–the render tree will be traversed and each node will be painted using the UI backend layer.
It's important to understand that this is a gradual process. For better user experience, the rendering engine will try to display contents on the screen as soon as possible. It will not wait until all HTML is parsed before starting to build and layout the render tree. Parts of the content will be parsed and displayed, while the process continues with the rest of the contents that keeps coming from the network.

DOM

The output tree (the "parse tree") is a tree of DOM element and attribute nodes. DOM is short for Document Object Model. It is the object presentation of the HTML document and the interface of HTML elements to the outside world like JavaScript.
The root of the tree is the "Document" object.

The DOM has an almost one-to-one relation to the markup. For example:

```
<html>
  <body>
    <p>
      Hello World
    </p>
    <div> <img src="example.png"/></div>
  </body>
</html>
```

This markup would be translated to the following DOM tree:

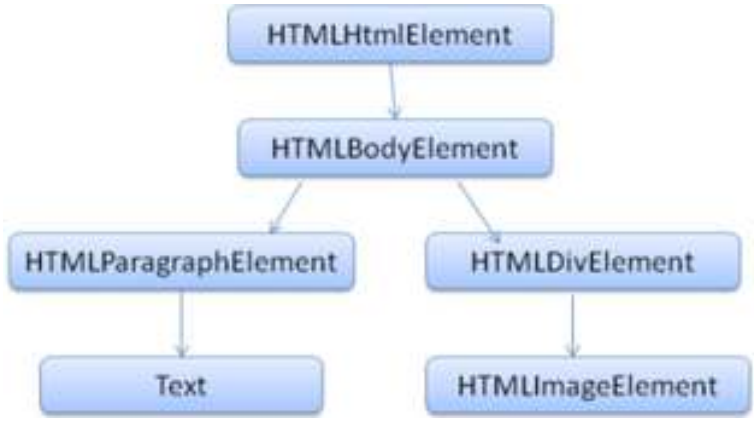

Figure : DOM tree of the example markup

Like HTML, DOM is specified by the W3C organization. See www.w3.org/DOM/DOMTR. It is a generic specification for manipulating documents. A specific module describes HTML specific elements. The HTML definitions can be found here: www.w3.org/TR/2003/REC-DOM-Level-2-HTML-20030109/idl-definitions.html.

When I say the tree contains DOM nodes, I mean the tree is constructed of elements that implement one of the DOM interfaces. Browsers use concrete implementations that have other attributes used by the browser internally.

The parsing algorithm

As we saw in the previous sections, HTML cannot be parsed using the regular top down or bottom up parsers.

The reasons are:

1. The forgiving nature of the language.
2. The fact that browsers have traditional error tolerance to support well known cases of invalid HTML.
3. The parsing process is reentrant. For other languages, the source doesn't change during parsing, but in HTML, dynamic code (such as script elements containing document.write() calls) can add extra tokens, so the parsing process actually modifies the input.

Unable to use the regular parsing techniques, browsers create custom parsers for parsing HTML.

The parsing algorithm is described in detail by the HTML5 specification. The algorithm consists of two stages: tokenization and tree construction.

Tokenization is the lexical analysis, parsing the input into tokens. Among HTML tokens are start tags, end tags, attribute names and attribute values.

The tokenizer recognizes the token, gives it to the tree constructor, and consumes the next character for recognizing the next token, and so on until the end of the input.

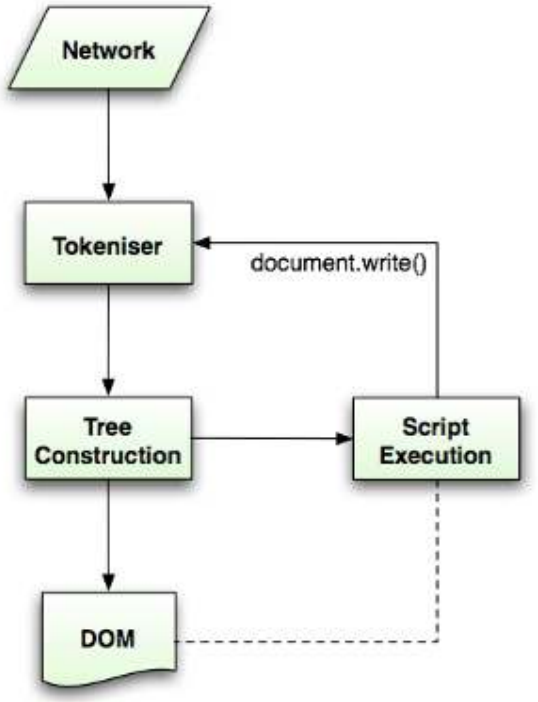

Figure : HTML parsing flow (taken from HTML5 spec)